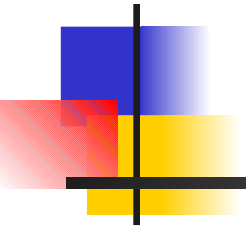


Player Tutorial

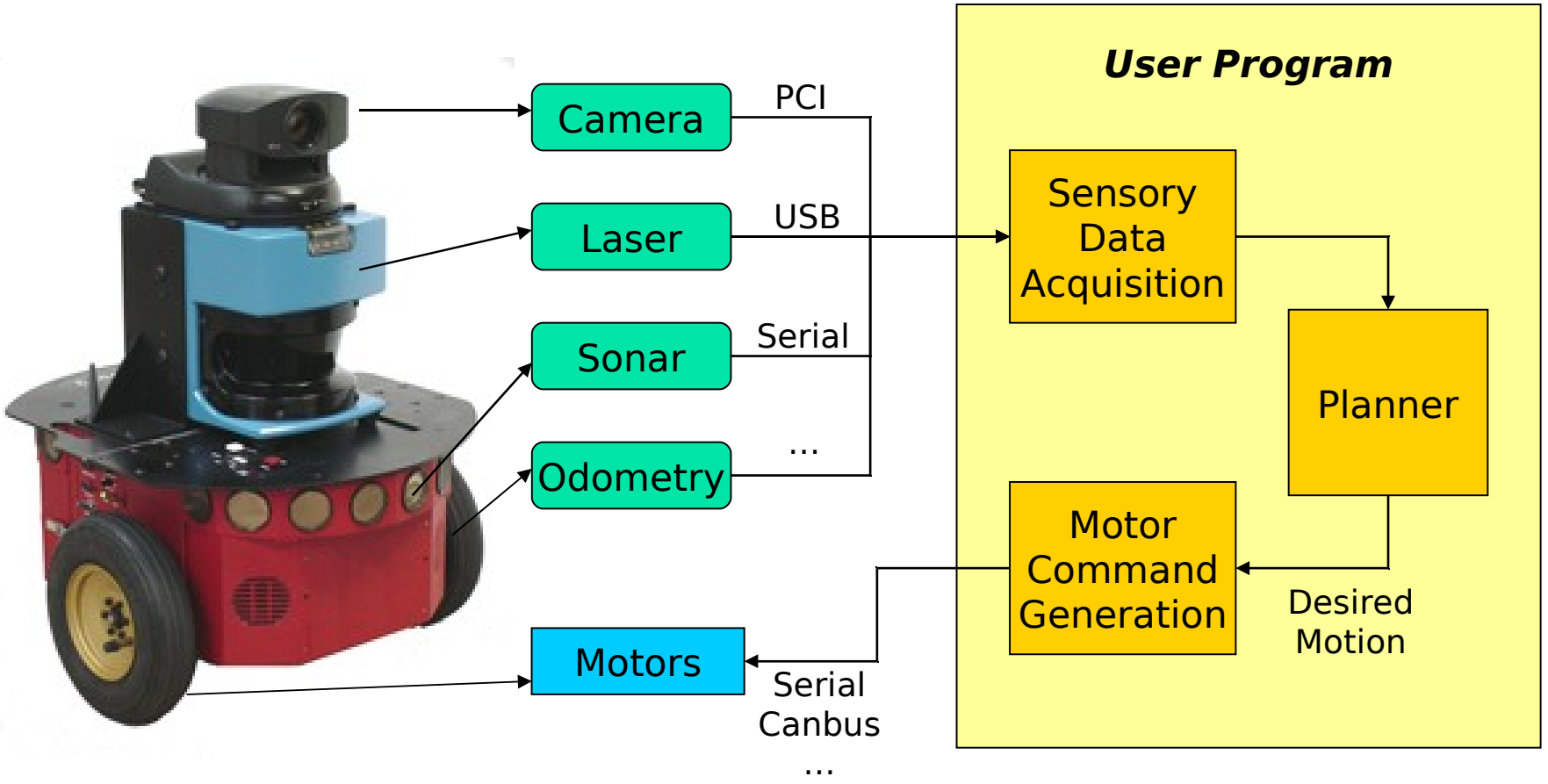
Boyoon Jung

Robotic Embedded Systems Lab
Robotics Research Lab
Center for Robotics and Embedded Systems

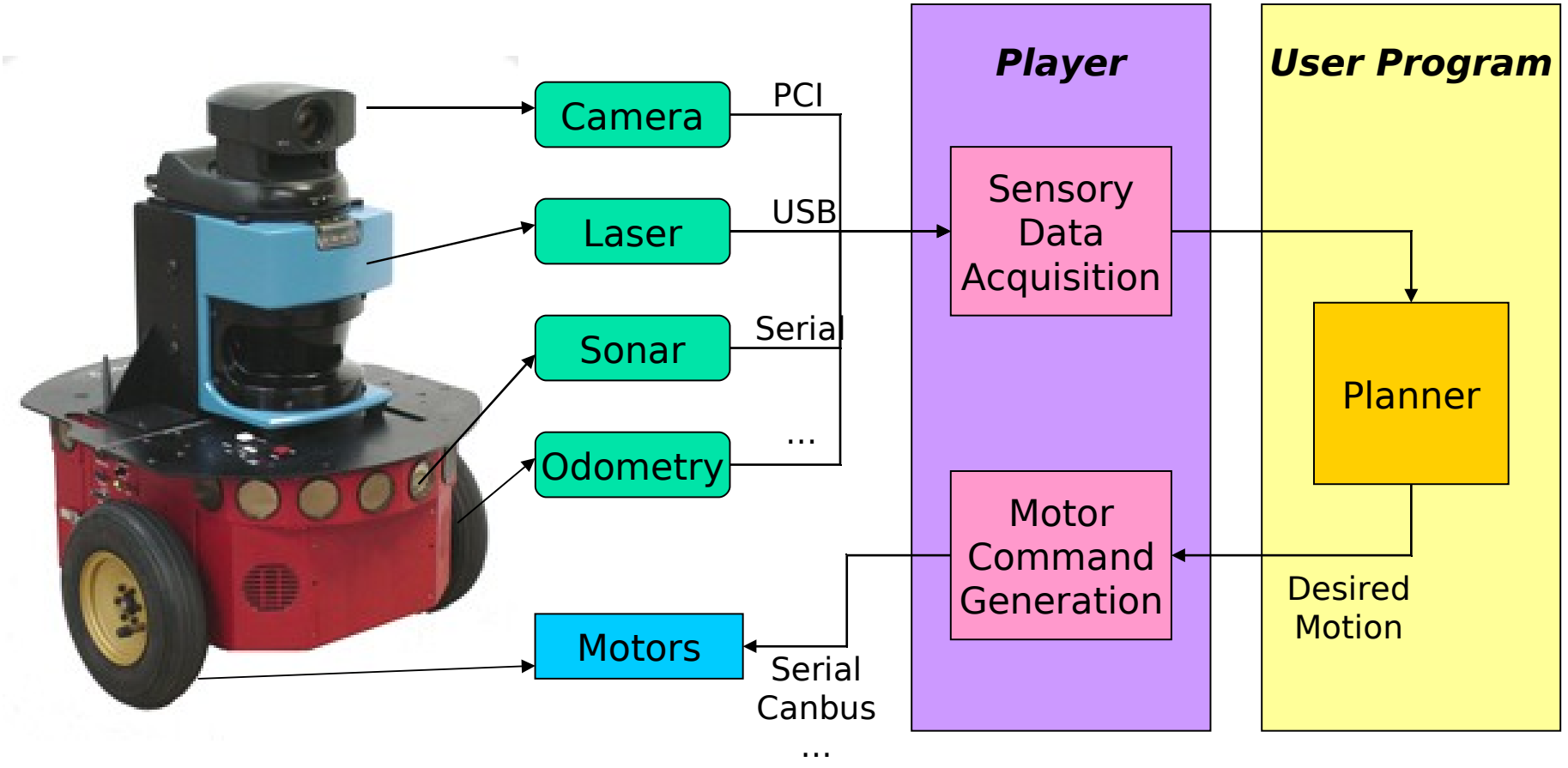
Player/Stage/Gazebo

- Player
  - Device server that provides a powerful, flexible interface to a variety of sensors and actuators
- Stage
  - 2D, multi-robot simulator
- Gazebo
  - 3D, dynamic, multi-robot simulator

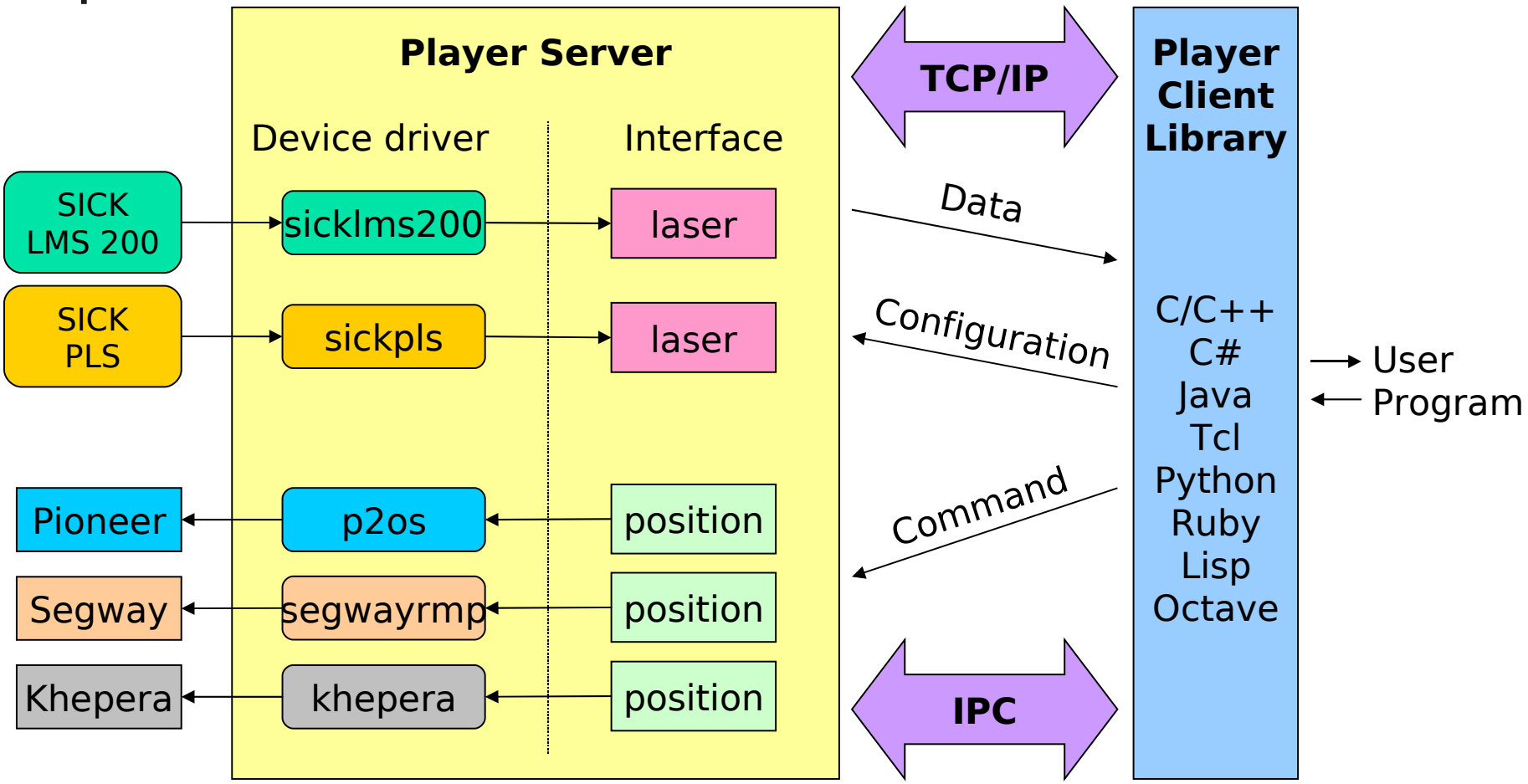
Robot Programming



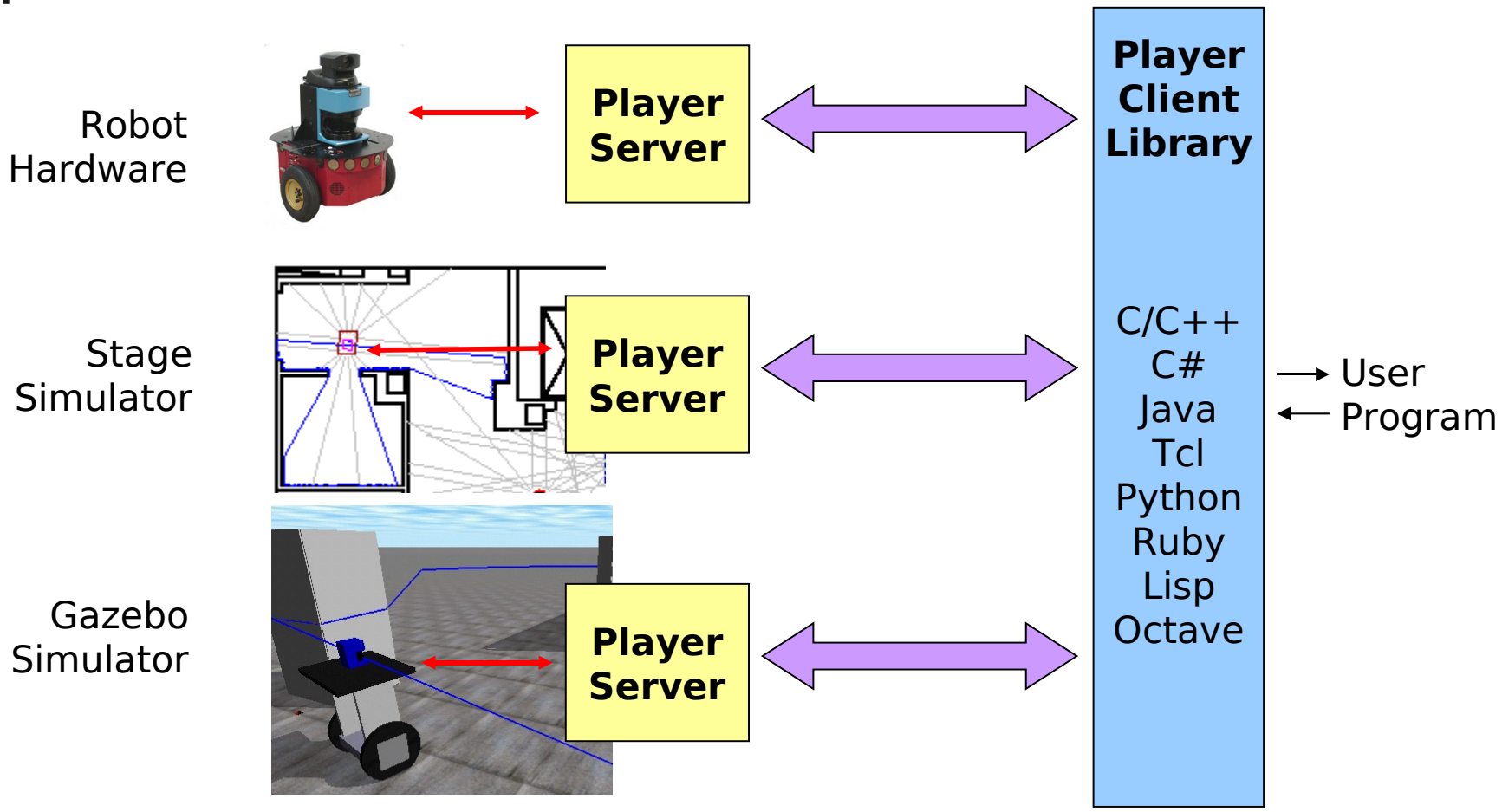
Player: a device server



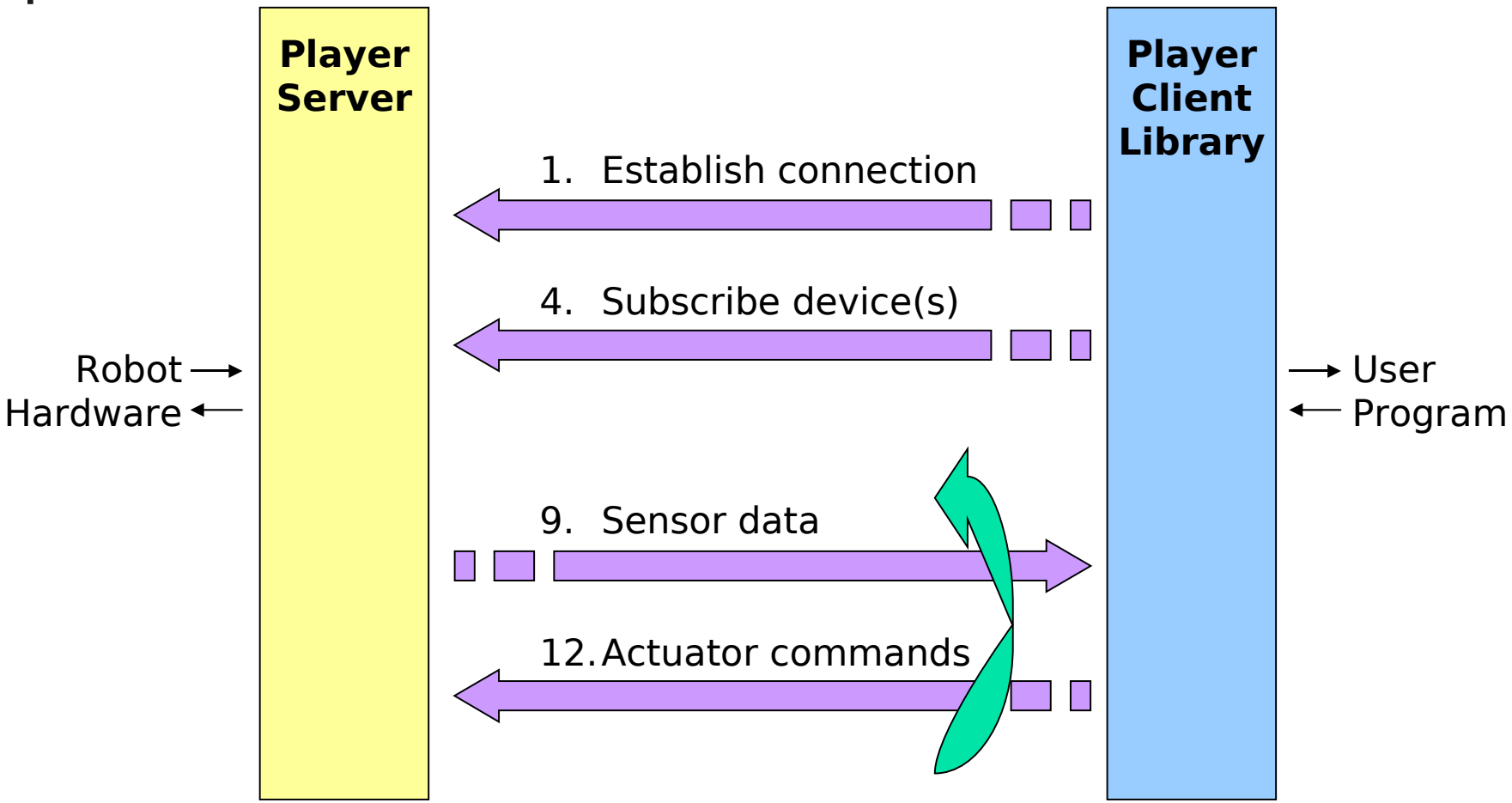
Client / Server Model



Hardware Abstraction



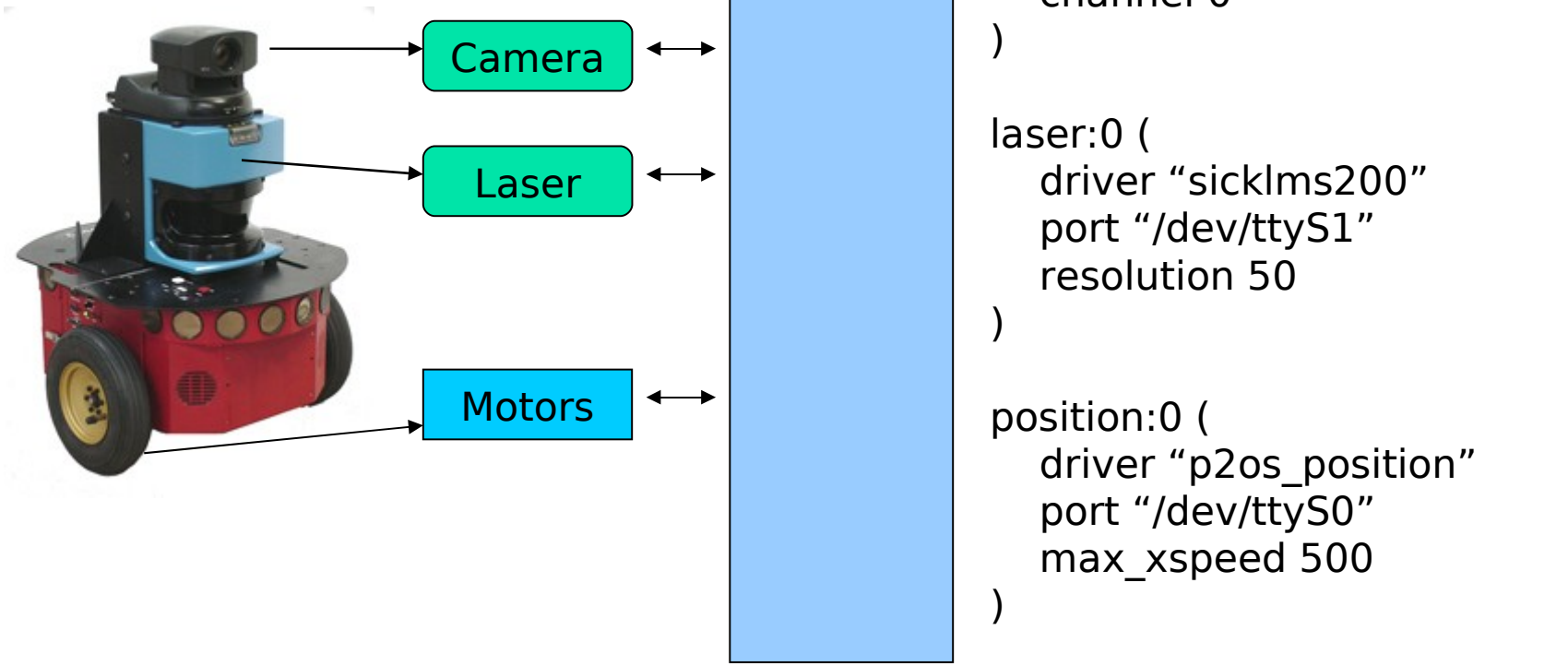
Communication



Running Player

- Building and installation
  - http://playerstage.sourceforge.net
  - `./configure`
  - `make`
  - `make install`
- Execution
  - `player [-p <port>] <config file>`
  - `playerv [hostname:port]`
  - `playerjoy [hostname:port]`

Player config file

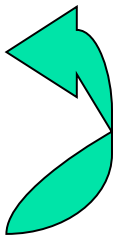



```
blobfinder:0 (
    driver “acts”
    devicepath=“/dev/video0”
    channel 0
)

laser:0 (
    driver “sicklms200”
    port “/dev/ttyS1”
    resolution 50
)

position:0 (
    driver “p2os_position”
    port “/dev/ttyS0”
    max_xspeed 500
)
```

Player Programming

- Player client libraries
  - C (libplayerc), C++ (libplayerc++), Python + wrappers for other languages

- Programming steps:
  1. Establish connection.
  2. Subscribe device(s).
  3. Read sensory data.
  4. Processing.
  5. Send motor command.



Repeat

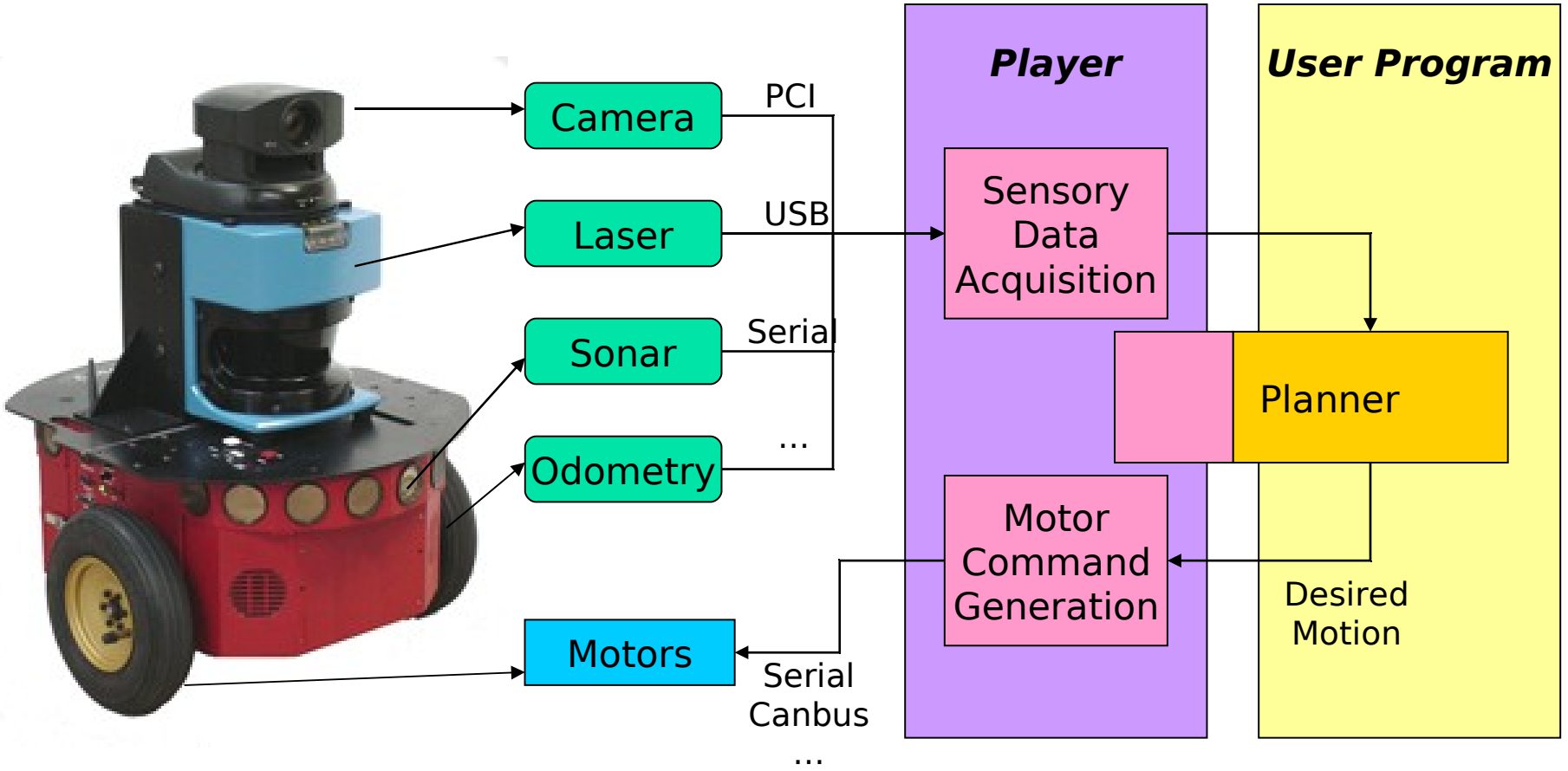
Example Code

- $PLAYER/examples/libplayerc++/laserobstacleavoid.cc

```
#include <libplayerc++/playerc++.h>

char host[256] = "localhost";
int port = 6665;
int index_pos = 0; int index_laser = 0;

int main(int argc, char **argv)
{
    // 1. Establish connection
    PlayerClient robot(host, port);

    // 2. Subscribe device(s)
    Position2dProxy pp(&robot, index_pos);
    LaserProxy lp(&robot, index_laser);
```

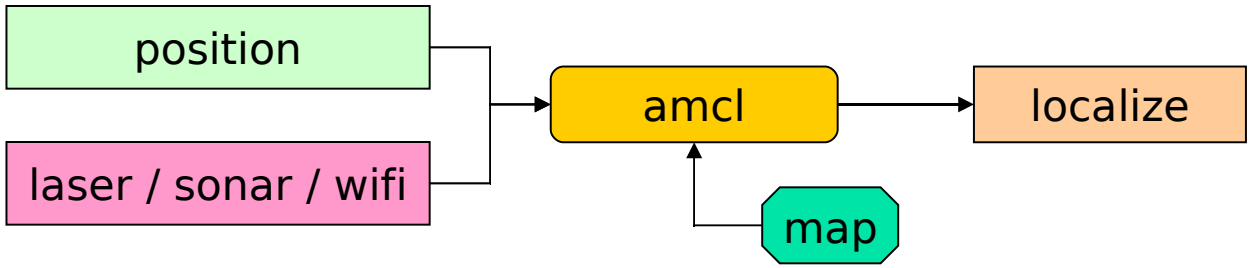
```
    pp.SetMotorEnable (true);
    for (;;)          // repeat
    {
       // 3. Read sensory data
       robot.Read();

       // 4. Procesing
       ...
       double l = (1e5*minR)/500 - 100;
       double r = (1e5*minL)/500 - 100;

       newspeed = (r+1)/1e3;
       newturnrate = (r-l);
       newturnrate = limit(newturnrate, -40,40);
       newturnrate = dtor(newturnrate);

       // 5. Send motor commands
       pp.SetSpeed(newspeed, newturnrate);
    }
}
```

Player: an algorithm repository

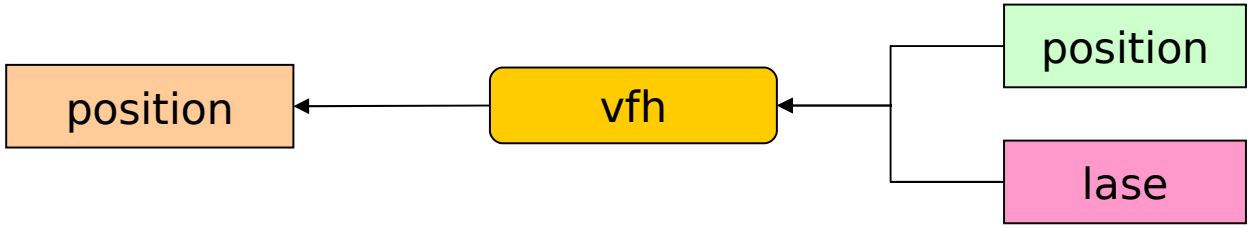


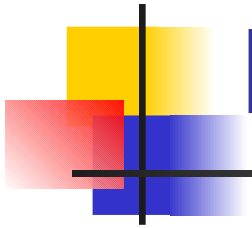
Computational devices

- amcl (Adaptive Monte-Carlo Localization)



- vfh (Vector Field Histogram)





Player

- A device server
  - Hardware abstraction
  - Network transparency
  - Language independence
- An algorithm repository